

CONTAINER SHIPPING SYSTEM LLD

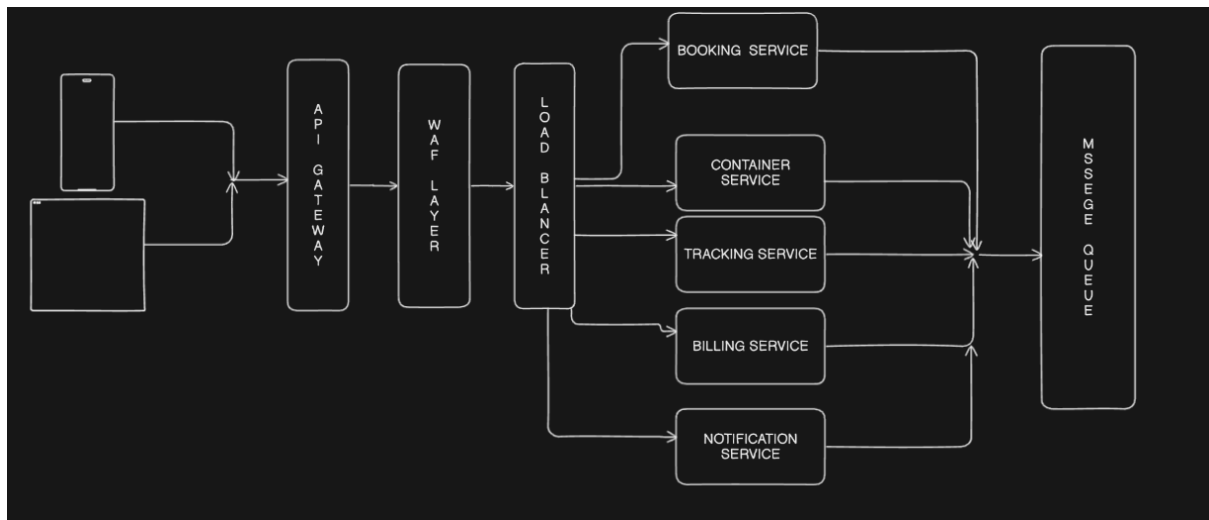
Document Information:

Field	Details
Document Version	V1.0
Created by	Patel Meet Bhaveshkumar
Date Created	07-08-2025
Status	Approved

Document Content:

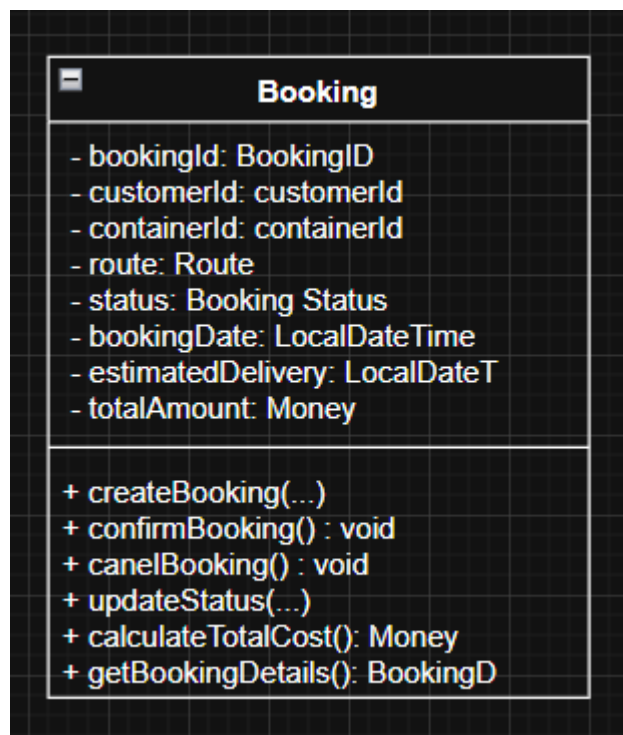
Sr No.	Description
1.	System Architecture Overview
2.	Class Diagram
3.	Component Diagrams
4.	Database Schema
5.	API Specification
6.	Design Pattern Applied
7.	Solid Principles Implementation
8.	Error Handling Strategy
9.	Performance Considerations

1. SYSTEM ARCHITECTURE OVERVIEW

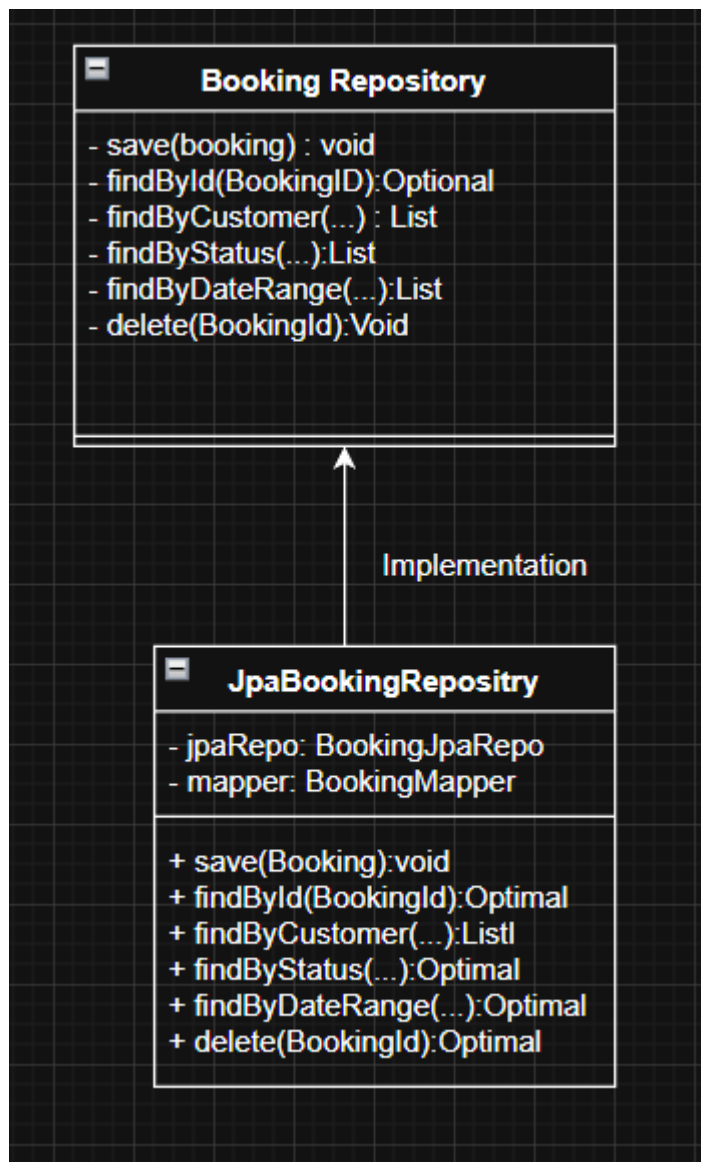


2. Class Diagram

2.1 Core Domain Classes

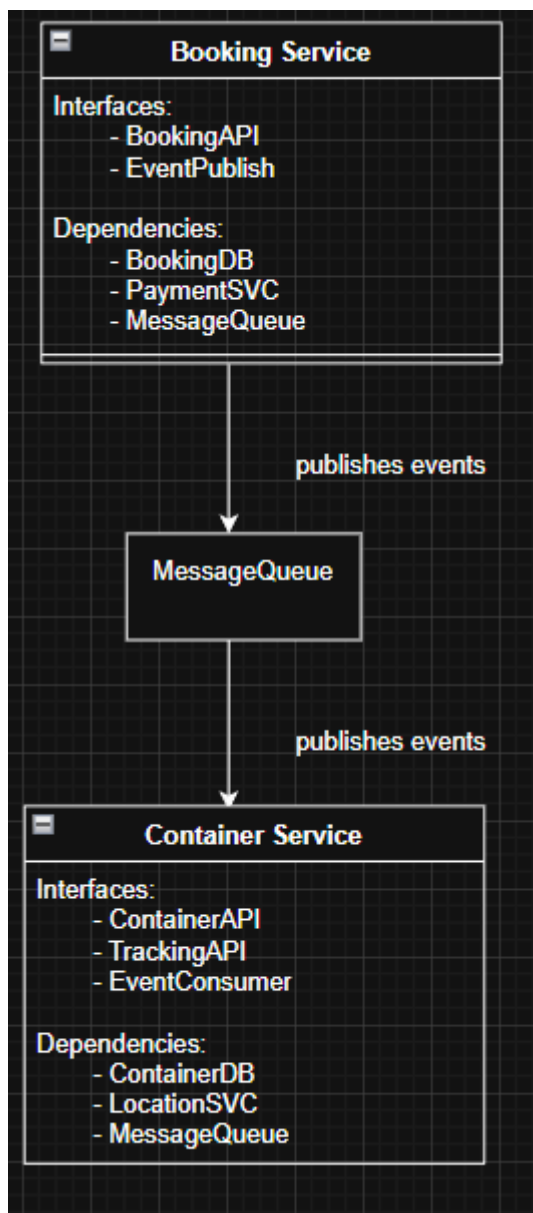


2. 3: Repository Pattern Implementation

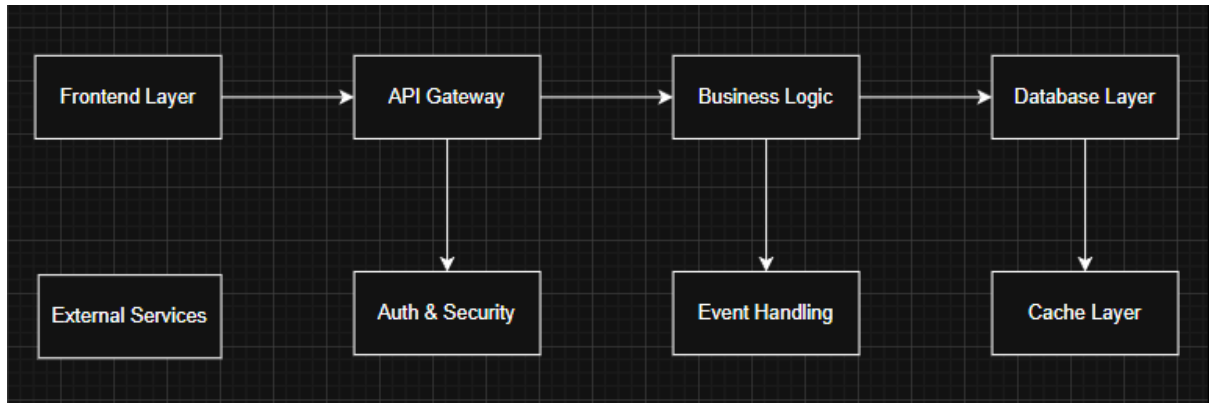


3.Component Diagrams

3.1 Microservices Architecture:



3.2 Data Flow Architecture:



4. Database Schema

4.1 Core Tables Structure

Bookings Table:

Table: Booking

Column Name	Data Type	Constraints	Description
Booking ID	VARCHAR(50)	PRIMARY KEY	Unique ID
Customer id	VARCHAR(50)	Not Null	Customer FK
Original port	VARCHAR(50)	Not Null	Origin Cost
Dest port	VARCHAR(50)	Not Null	Dest Code
Status	VARCHAR(50)	Not Null	Enum status
booking_date	TIMESTAMP	Not Null	Created AT
estimated del	TIMESTAMP	Null	ETA
total amount	DECIMAL(10,2)	Not Null	Total Cost
created AT	TIMESTAMP	DEFAULT NOW	Created
Updated_AT	TIMESTAMP	ON UPDATE	Modified

Indexes: - idx_customer_id (customer_id)

- idx_booking_date (booking_date)

- idx_status (status)

Containers Table:

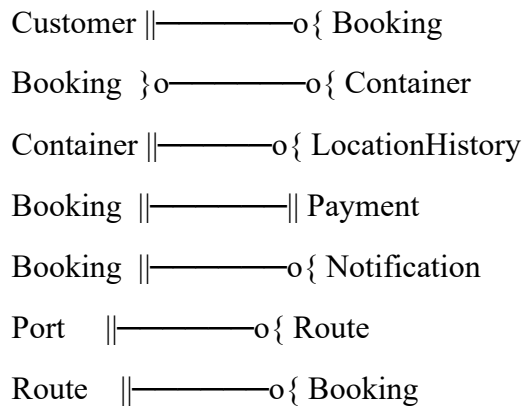
Column Name	Data Type	Constraints	Description
Customer id	VARCHAR(50)	PRIMARY KEY	Unique ID
Type	VARCHAR(50)	Not Null	Type enum
Status	VARCHAR(50)	Not Null	Status Enum
Current_lat	DECIMAL(9,6)	Null	Lastitude

Current_lng	DECIMAL(9,6)	Null	Logitude
Weight	DECIMAL(9,6)	DEFAULT 0	Weight
Contents	TEXT	Null	Description
Last_updated	TIMESTAMP	DEFAULT NOW	Last Updated
Created_at	TIMESTAMP	DEFAULT NOW	Created

Booking-Container Mapping:

Column Name	Data Type	Constraints	Description
id	BIGINT	PRIMARY KEY	Auto Inc
Booking_ID	VARCHAR(50)	FK, NOT NULL	Booking Ref
Container_id	VARCHAR(50)	FK, NOT NULL	Container
Assigned_at	DECIMAL(9,6)	Default Now	Assignment

4.2 Entity Relationships



5. API Specifications

5.1 Booking API Endpoints:

Create Booking:

POST /api/v1/bookings

Content-Type: application/json

Authorization: Bearer {token}

Request Body:

```
{
  "customerId": "string",
  "originPort": "string",
  "destinationPort": "string",
  "containerRequirements": [
    {
      "type": "STANDARD_20FT",
      "quantity": 2,
      "contents": "string"
    }
  ],
  "preferredDate": "2024-01-15T10:00:00Z",
  "specialRequirements": "string"
}
```

Response (201 Created):

```
{
  "bookingId": "BK_2024_001234",
  "status": "PENDING",
  "estimatedCost": 1250.00,
  "estimatedDelivery": "2024-01-25T14:00:00Z",
  "message": "Booking created successfully"
}
```

Error Response (400 Bad Request):

```
{
  "error": "INSUFFICIENT_CONTAINERS",
  "message": "Not enough containers available",
  "details": {
    "requested": 5,
    "available": 3
  }
}
```

Get Booking Details:

GET /api/v1/bookings/{bookingId}

Authorization: Bearer {token}

Response (200 OK):

```
{
  "bookingId": "BK_2024_001234",
  "customerId": "CUST_001",
  "status": "CONFIRMED",
  "route": {
    "origin": "MUMBAI",
    "destination": "DUBAI",
    "distance": 1200.5
  },
  "containers": [
    {
      "containerId": "CONT_123456",
      "type": "STANDARD_20FT",
      "currentStatus": "IN_TRANSIT",
      "currentLocation": {
        "latitude": 19.0760,
        "longitude": 72.8777,
        "timestamp": "2024-01-20T08:30:00Z"
      }
    }
  ]
}
```



```
    }  
  ],  
  "totalAmount": 1250.00,  
  "paymentStatus": "COMPLETED"  
}
```

5.2 Container Tracking API:

Track Container:

GET /api/v1/containers/{containerId}/track

Authorization: Bearer {token}

Response (200 OK):

```
{  
  "containerId": "CONT_123456",  
  "currentLocation": {  
    "latitude": 19.0760,  
    "longitude": 72.8777,  
    "address": "Mumbai Port, India",  
    "timestamp": "2024-01-20T08:30:00Z"  
  },  
  "status": "IN_TRANSIT",  
  "locationHistory": [  
    {  
      "latitude": 18.9220,  
      "longitude": 72.8347,  
      "address": "Departure Port",  
      "timestamp": "2024-01-15T10:00:00Z"  
    }  
  ],  
  "estimatedArrival": "2024-01-25T14:00:00Z"  
}
```

6. Design Patterns Applied:

6.1 Factory Pattern

ContainerFactory

└─ createStandardContainer()
└─ createRefrigeratedContainer()
└─ createHighCubeContainer()
└─ createTankContainer()

Usage Context:

- Container creation based on booking requirements
- Centralized container instantiation logic
- Type-specific initialization handling

6.2 Strategy Pattern:

PricingStrategy Interface

└─ DistanceBasedPricing
└─ WeightBasedPricing
└─ VolumeBasedPricing
└─ PremiumServicePricing

Context: BookingService

- Flexible pricing calculation
- Easy addition of new pricing models
- Runtime strategy selection

6.3 Observer Pattern:

Event Publishing System:

BookingCreated ─┐
ContainerMoved ─┐
PaymentCompleted ─┐ ──> EventBus ──> Multiple Listeners
DeliveryComplete ─┐ ──> NotificationService

StatusChanged —┐

└─▶ BillingService

└─▶ AnalyticsService

6.4 Repository Pattern

Repository Abstraction Layer:

Domain Model ←──→ Repository Interface ←──→ Data Access
Implementation

Booking ←──→ BookingRepository ←──→ JpaBookingRepository

Container ←──→ ContainerRepository ←──→ JpaContainerRepository

7. SOLID Principles Implementation:

7.1 Single Responsibility Principle

- BookingService: Only handles booking operations
- ContainerService: Only manages container lifecycle
- PaymentService: Only processes payments
- NotificationService: Only sends notifications

7.2 Open/Closed Principle

- PricingStrategy: Open for extension (new pricing models), closed for modification
- NotificationChannel: Can add new channels without changing existing code
- ContainerType: Extensible for new container types

7.3 Liskov Substitution Principle

- Container hierarchy: All container types can be used interchangeably
- Repository implementations: Any repository implementation works with service layer

7.4 Interface Segregation Principle

- Separate interfaces for different concerns:
 - ContainerTracker for location updates
 - ContainerInventory for availability management
 - ContainerMaintenance for maintenance operations

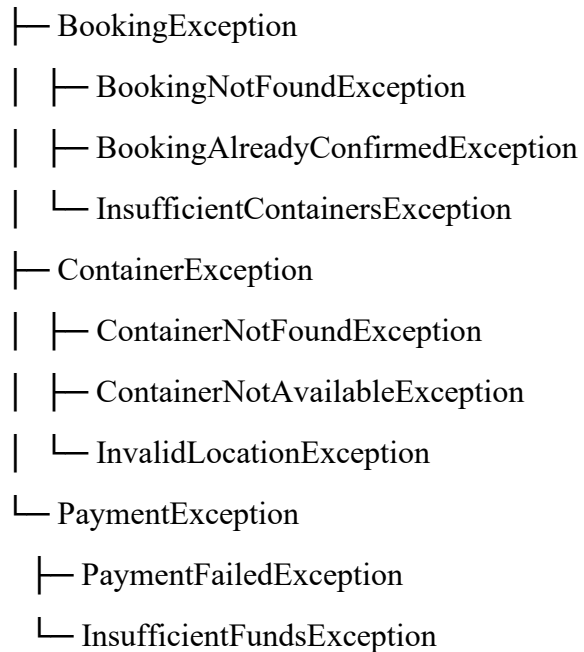
7.5 Dependency Inversion Principle

- High-level modules (Services) depend on abstractions (Interfaces)
- Low-level modules (Repositories) implement the abstractions

- Dependency injection used throughout the system
-

8. Error Handling Strategy:

8.1 Exception Hierarchy:



8.2 Global Error Handling:

HTTP Status Code Mapping:

400 Bad Request	←— Validation Errors
401 Unauthorized	←— Authentication Issues
404 Not Found	←— Resource Not Found
409 Conflict	←— Business Rule Violations
422 Unprocessable	←— Business Logic Errors
500 Internal Error	←— System Errors

9. Performance Considerations

9.1 Caching Strategy

- Redis Cache: Frequently accessed booking details
- Application Cache: Container availability data
- CDN: Static content and images

9.2 Database Optimization

- Read Replicas: For tracking and reporting queries
- Indexing: On frequently queried columns
- Partitioning: Large tables by date/region

9.3 Monitoring and Metrics

- Response Time: API endpoint performance
- Throughput: Requests per second
- Error Rate: Failed requests percentage
- Resource Usage: CPU, Memory, Database connections